Home.java:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;



public class home extends JFrame{

    // form components

    private JLabel w1, w2, image;

    private JButton reg, log;



    private JLabel emailLabel, passwordLabel, login;

    private JTextField emailField;

    private JPasswordField passwordField;

    private JButton loginButton, clearButton;
```

```
    public home() {

        setTitle("Integrity Auditing");

        setSize(getMaximumSize());

        setLocationRelativeTo(null);



        /*image = new JLabel();

        image.setIcon(new ImageIcon("C:\\Users\\Shivam\\Desktop\\main project\\B13\\img\\home.jpg")); //Sets the image to be displayed as an icon

        Dimension size = image.getPreferredSize(); //Gets the size of the image

        image.setBounds(350, 100, 600, 500);

*/

        JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));

        panel.setBackground(Color.LIGHT_GRAY);



        JLabel w1 = new JLabel("Enabling Integrity Auditing Based on the Keyword With Sensitive Information Privacy for Encrypted Cloud Data");
```

```
w1.setFont(new Font("Times New Roman",Font.BOLD,23));

w1.setPreferredSize(new Dimension(1150,100));



JLabel w2 = new JLabel("Login");

w2.setFont(new Font("Times New Roman",Font.BOLD,20));

w2.setHorizontalAlignment(SwingConstants.CENTER);

w2.setPreferredSize(new Dimension(1150,100));



JButton reg = new JButton("Register");

reg.setPreferredSize(new Dimension(400,30));

reg.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        register reg = new register();

        reg.setVisible(true);

        setVisible(false);
```

```
    }
});
JButton log = new JButton("Login");
log.setPreferredSize(new Dimension(400,30));
log.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        login log = new login();
        log.setVisible(true);
        setVisible(false);
    }
});
panel.add(w1);
panel.add(Box.createRigidArea(new Dimension(10, 0)));
panel.add(w2);
//add(image);
```

```
        panel.add(reg);
        panel.add(log);


        add(panel);
        setVisible(true);
    }
    public static void main(String args[]){
        new home();
    }
}
```

Register.java:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
```

```
public class register extends JFrame implements ActionListener {
    // form components
    private JLabel reg, nameLabel, emailLabel, passwordLabel, confirmLabel;
    private JTextField nameField, emailField;
    private JPasswordField passwordField, confirmField;
    private JButton registerButton, clearButton, cancelButtuon;

    public register() {
        setTitle("Registration Form");
        setBackground(new Color(124,230,12));
        // create components
        reg = new JLabel("Register");
        reg.setFont(new Font("Times New Roman",Font.BOLD,26));
        nameLabel = new JLabel("Name:");
        emailLabel = new JLabel("Email:");
        passwordLabel = new JLabel("Password:");
```

```
confirmLabel = new JLabel("Confirm Password:");

nameField = new JTextField(20);

emailField = new JTextField(20);

passwordField = new JPasswordField(20);

confirmField = new JPasswordField(20);

registerButton = new JButton("Register");

clearButton = new JButton("Clear");

cancelButtuon = new JButton("back");


// set layout

setLayout(new GridBagLayout());

setBackground(Color.MAGENTA);

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(5, 5, 5, 5);

gbc.gridx = 1;

gbc.gridy = 0;
```

```
add(reg, gbc);

gbc.gridx = 0;

gbc.gridy = 1;

add(nameLabel, gbc);

gbc.gridx = 1;

gbc.gridy = 1;

add(nameField, gbc);

gbc.gridx = 0;

gbc.gridy = 2;

add(emailLabel, gbc);

gbc.gridx = 1;

gbc.gridy = 2;

add(emailField, gbc);

gbc.gridx = 0;

gbc.gridy = 3;

add(passwordLabel, gbc);
```

```
gbc.gridx = 1;

gbc.gridy = 3;

add(passwordField, gbc);

gbc.gridx = 0;

gbc.gridy = 4;

add(confirmLabel, gbc);

gbc.gridx = 1;

gbc.gridy = 4;

add(confirmField, gbc);

gbc.gridx = 0;

gbc.gridy = 5;

add(registerButton, gbc);

gbc.gridx = 1;

gbc.gridy = 5;

add(clearButton, gbc);

gbc.gridx = 2;
```

```
    gbc.gridy = 5;
    add(cancelButtuon, gbc);

    // add action listeners
    registerButton.addActionListener(this);
    clearButton.addActionListener(this);
    cancelButtuon.addActionListener(this);

    // set size and visibility
    setSize(getMaximumSize());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == registerButton) {
```

```
        // validate form input
        String name = nameField.getText().trim();
        String email = emailField.getText().trim();
        char[] password = passwordField.getPassword();
        char[] confirm = confirmField.getPassword();
        if (name.isEmpty() || email.isEmpty() || password.length == 0 ||
confirm.length == 0) {
            JOptionPane.showMessageDialog(this, "Please fill out all fields.");
        } else if (!String.valueOf(password).equals(String.valueOf(confirm))) {
            JOptionPane.showMessageDialog(this, "Passwords do not match.");
        } else {
            // write data to CSV file
            try {
                BufferedWriter writer = new BufferedWriter(new
FileWriter("users.csv", true));
                writer.write(name + "," + email + "," + String.valueOf(password));
                writer.newLine();
```

```
                writer.close();
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(this, "Error writing to file: " +
ex.getMessage());
            }
            JOptionPane.showMessageDialog(this, "Registration successful.");
            nameField.setText("");
            emailField.setText("");
            passwordField.setText("");
            confirmField.setText("");
            login log = new login();
            log.setVisible(true);
            setVisible(false);
        }
    } else if (e.getSource() == clearButton) {
        // clear form input
        nameField.setText("");
```

```
            emailField.setText("");
            passwordField.setText("");
        }else if (e.getSource()==cancelButtuon){
            JOptionPane.showMessageDialog(this, "   Back to Home Pgae");
            home hm = new home();
            hm.setVisible(true);
            setVisible(false);
        }
    }
    public static void main(String args[]){
        new register();
    }
}
```

Login.java:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.io.*;


public class login extends JFrame implements ActionListener {

    // form components

    private JLabel emailLabel, passwordLabel, image, login;

    private JTextField emailField;

    private JPasswordField passwordField;

    private JButton loginButton, clearButton;


    public login() {

        setTitle("Login Form");


        // create components
```

```
login = new JLabel("Login");

login.setFont(new Font("Times New Roman",Font.BOLD,26));

emailLabel = new JLabel("Email:");

passwordLabel = new JLabel("Password:");

emailField = new JTextField(20);

passwordField = new JPasswordField(20);

loginButton = new JButton("Login");

clearButton = new JButton("Cancel");

// set layout

setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(5, 5, 5, 5);

gbc.gridx = 1;

gbc.gridy = 0;

add(login, gbc);

gbc.gridx = 0;
```

```
gbc.gridy = 1;

add(emailLabel, gbc);

gbc.gridx = 1;

gbc.gridy = 1;

add(emailField, gbc);

gbc.gridx = 0;

gbc.gridy = 2;

add(passwordLabel, gbc);

gbc.gridx = 1;

gbc.gridy = 2;

add(passwordField, gbc);

gbc.gridx = 0;

gbc.gridy = 3;

add(loginButton, gbc);

gbc.gridx = 1;

gbc.gridy = 3;
```

```
    add(clearButton,gbc);


    // add action listeners
    loginButton.addActionListener(this);
    clearButton.addActionListener(this);


    // set size and visibility
    setSize(getMaximumSize());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}


public void actionPerformed(ActionEvent e) {
    if (e.getSource() == loginButton) {
        // validate form input
        String email = emailField.getText().trim();
```

```
        char[] password = passwordField.getPassword();
        if (email.isEmpty() || password.length == 0) {
            JOptionPane.showMessageDialog(this, "Please enter your email and password.");
        } else {
            // check user credentials against CSV file
            boolean found = false;
            try {
                FileReader reader = new FileReader("users.csv");
                BufferedReader bufferedReader = new BufferedReader(reader);
                String line;
                while ((line = bufferedReader.readLine()) != null) {
                    String[] parts = line.split(",");
                    if (parts[1].equals(email) && parts[2].equals(String.valueOf(password))) {
                        found = true;
                        break;
```

```
            }
        }
        bufferedReader.close();
        reader.close();
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
    }
    if (found) {
        JOptionPane.showMessageDialog(this, "Login Successfully");
        Main M1 = new Main();
        M1.setVisible(true);
        setVisible(false);
    } else {
        JOptionPane.showMessageDialog(this, "Invalid email or password.");
    }
}
```

```
        } else if (e.getSource() == clearButton) {
            // clear form input
            emailField.setText("");
            passwordField.setText("");
            home h = new home();
            h.setVisible(true);
            setVisible(false);
        }
    }
}
```

Main.java:

```
import it.unisa.dia.gas.jpbc.Element;
import it.unisa.dia.gas.jpbc.Pairing;
import it.unisa.dia.gas.jpbc.PairingParametersGenerator;
import it.unisa.dia.gas.plaf.jpbc.pairing.PairingFactory;
import it.unisa.dia.gas.plaf.jpbc.pairing.e.TypeECurveGenerator;
```

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.math.BigInteger;



/*

    Checking only when it is necessary: Enabling integrity auditing based

    on the keyword with sensitive information privacy for encrypted cloud data

 */

public class Main extends JFrame {

    JPanel panel;

    JLabel l1, l2, l3, l4;

    JButton b1, b2, b3, b4;

    public Main(){
```

```
setTitle("Integrity ");

setTitle("Welcome");

setSize(getMaximumSize());

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


JLabel w1 = new JLabel(" Enabling Integrity Auditing Based on the Keyword With Sensitive Information Privacy for Encrypted Cloud Data");

w1.setFont(new Font("Times New Roman",Font.BOLD,24));

w1.setPreferredSize(new Dimension(1250,100));


JLabel m=new JLabel("Output");

m.setFont(new Font("Times New Roman",Font.BOLD,40));

m.setPreferredSize(new Dimension(150, 50));

m.setFont(new Font("Times New Roman",Font.BOLD,18));


JTextArea m1=new JTextArea();
```

```
m1.setPreferredSize(new Dimension(1155, 500));

m1.setBorder(BorderFactory.createLineBorder(Color.BLACK,1));



JButton bt1 = new JButton("System Initialization");

bt1.setPreferredSize(new Dimension(150,40));



JButton bt2 = new JButton("Setup");

bt2.setPreferredSize(new Dimension(150,40));



JButton bt3 = new JButton("Index generation");

bt3.setPreferredSize(new Dimension(150,40));



JButton bt4 = new JButton("Authenticator generation");

bt4.setPreferredSize(new Dimension(150,40));



JButton bt5 = new JButton("Trapdoor generation");
```

```
bt5.setPreferredSize(new Dimension(150,40));

JButton bt6 = new JButton("Challenge generation");

bt6.setPreferredSize(new Dimension(150,40));

JButton bt7 = new JButton("Proof generation");

bt7.setPreferredSize(new Dimension(150,40));

JButton bt8 = new JButton("Proof Verification");

bt8.setPreferredSize(new Dimension(150,40));

JPanel panel=new JPanel();

panel.setBackground(Color.LIGHT_GRAY);

panel.add(w1);

panel.add(m);
```

```
        panel.add(m1);

        panel.add(bt1);

        panel.add(bt2);

        panel.add(bt3);

        panel.add(bt4);

        panel.add(bt5);

        panel.add(bt6);

        panel.add(bt7);

        panel.add(bt8);



        add(panel);

        setVisible(true);
// code starts here

        int rBits = 160;

        int qBits = 512;
```

```
PairingParametersGenerator pg = new TypeECurveGenerator(rBits, qBits);

Pairing pairing = PairingFactory.getPairing("params.properties");

PairingFactory.getInstance().setUsePBCWhenPossible(true);



/*

    1. System initialization

        q order, G1, G2 are already set in the params.properties file.

        e: G1 * G1 -> G2   This pairing function is also already set in the
param.properties file

        Two generators u, g are selected here from the elliptic curve group G1

        Three secure hash functions - yet to be decided

        Symmetric encryption algorithm - to use the existing code from the
Internet

        Pseudorandom keys - yet to be decided

        secret key x chosen from Zq* here

        public key y = g power x computed here

*/
```

```
System.out.println("\n\n 1.System Initialization");

Element u = pairing.getG1().newRandomElement().getImmutable();

Element g = pairing.getG1().newRandomElement().getImmutable();

Element x = pairing.getZr().newRandomElement().getImmutable();

Element y = g.powZn(x);

System.out.println("u is : " + u);

System.out.println("g is : " + g);

System.out.println("x is : " + x);

System.out.println("y is : " + y);

bt1.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        m1.setText("\nU is :"+u.toString()+"\n\nG is :"+g.toString()+"\n\nX is
:"+x.toString()+"\n\nY is :"+y.toString());

        m1.setFont(new Font("Times New
Roman",Font.TRUETYPE_FONT,16));

        m1.setLineWrap(true);
```

```
    }
});
/* Setup phase - starts here */


/* Step 1: For each file, the user splits the file into s blocks.
        and encrypts them by the symmetric key algorithm (e.g. AES) */
// Let us assume our file has 3 blocks such as b1, b2, b3. Let us randomly select the block values of the file from zq*
System.out.println("\n\n 2.Setup");
Element b1 = pairing.getZr().newRandomElement().getImmutable();
Element b2 = pairing.getZr().newRandomElement().getImmutable();
Element b3 = pairing.getZr().newRandomElement().getImmutable();
System.out.println("This our actual file blocks");
System.out.println("b1 is:" + b1);
System.out.println("b2 is:" + b2);
System.out.println("b3 is:" + b3);
```

```
// The blocks b1, b2, b3 are encrypted using AES algorithm.\

// Let us assume that, the encrypted blocks for b1, b2, b3 are c1, c2, c3.

// For now, let us randomly select the values of c1, c2, c3.


Element c1 = pairing.getZr().newRandomElement().getImmutable();

Element c2 = pairing.getZr().newRandomElement().getImmutable();

Element c3 = pairing.getZr().newRandomElement().getImmutable();

System.out.println("\nThis our file Encrypted blocks");

System.out.println("c1 is:" + c1);

System.out.println("c2 is:" + c2);

System.out.println("c3 is:" + c3);


/* Step 2: The user extracts all thaaae keywords, then builds the keyword set
W.

Let us assume that, we have three keywords. It is represented using an array.
*/

System.out.println("\nThis our keywords");
```

```
Element k0 = pairing.getZr().newRandomElement().getImmutable();

Element k1 = pairing.getZr().newRandomElement().getImmutable();

Element k2 = pairing.getZr().newRandomElement().getImmutable();

Element keyword[] = new Element[3];

keyword[0] = k0;

keyword[1] = k1;

keyword[2] = k2;

System.out.println("keyword[0] is:" + keyword[0]);

System.out.println("keyword[1] is:" + keyword[1]);

System.out.println("keyword[2] is:" + keyword[2]);

/* Step 3 :
```

For each keyword wk, the user creates an n-bit binary string as the index vector vwk.

He initializes every element this index vector to 0.

For each file Fi, if it contains keyword wk, then the user sets the i-th bit of the index vector to 1: vwk[i] = 1

```
        In this case, we assume that, indexvector[3].

        Indexvector[0] = 1 means, keyword[0] is present in file 1.

        Indexvector[1] = 1 means, keyword[1] is present in file 1.

        Indexvector[2] = 1 means, keyword[2] is present in file 1.

     */

    Element indexvector[] = new Element[3];

    indexvector[0] = pairing.getZr().newElement(new BigInteger("1"));

    // indexvector[0] is vw0

    indexvector[1] = pairing.getZr().newElement(new BigInteger("1"));

    // indexvector[1] is vw1

    indexvector[2] = pairing.getZr().newElement(new BigInteger("1"));

    // indexvector[2] is vw2

    bt2.addActionListener(new ActionListener() {

        @Override

        public void actionPerformed(ActionEvent e) {

            m1.setText("\nb1 is :"+b1.toString()+"\n\nb2 is :"+b2.toString()+"\n\nb3
is :"+b3.toString()+"\n\nc1 is :"+c1.toString()+"\n\nc2 is :"+c2.toString()+"\n\nc3
```

```
is :"+c3.toString()+"\n\nkeyword[0] is :"+keyword[0].toString()+"\n\nkeyword[1]
is :"+keyword[1].toString()+"\n\nkeyword[2] is :"+keyword[2].toString());

            m1.setLineWrap(true);

        }

    });



    /* Setup phase - ends here */



    /* Index generation - starts here */

    System.out.println("\n\n 3.Index Generation");

    // step 1: For each keyword wk, the user computes pi(wk) as

    //         the address of each row in the secure index.

    System.out.println("Step 1:");

    /* In the future, if the code correctly works as per the research manuscript,

       this pik0 should be replaced by pik0 = pi(k0)

       LLLrly for pik1, pik2 also.

     */
```

```
        Element pik0 = pairing.getZr().newRandomElement().getImmutable();

        Element pik1 = pairing.getZr().newRandomElement().getImmutable();

        Element pik2 = pairing.getZr().newRandomElement().getImmutable();

        System.out.println("pik0 is:" + pik0);

        System.out.println("pik1 is:" + pik1);

        System.out.println("pik2 is:" + pik2);

        // step 2: For each keyword wk, the index vector is encrypted using

        //        the exclusive or of vwk and f(pi(wk)).

        //        In future, fpik0 will be replaced with a suitable pseudo-random
function.

        System.out.println("Step 2:");

        Element fpik0 = pairing.getZr().newRandomElement().getImmutable();

        Element fpik1 = pairing.getZr().newRandomElement().getImmutable();

        Element fpik2 = pairing.getZr().newRandomElement().getImmutable();



        Element evpiwk0 = pairing.getZr().newElement(new
BigInteger(element_xor(indexvector[0].toString(), fpik0.toString())));
```

```
        System.out.println("evpiwk0 is : " + evpiwk0);

        Element evpiwk1 = pairing.getZr().newElement(new
BigInteger(element_xor(indexvector[1].toString(), fpik1.toString())));

        System.out.println("evpiwk1 is : " + evpiwk1);

        Element evpiwk2 = pairing.getZr().newElement(new
BigInteger(element_xor(indexvector[2].toString(), fpik2.toString())));

        System.out.println("evpiwk2 is : " + evpiwk2);

        // String string_indexvector0 = indexvector[0].toString();

        // v1 = pairing.getZr().newElement(new BigInteger(""));




        //int_indexvector[0] = indexvector[0].toBigInteger();



        /* int a = 7;

        int b = 4;

        int c = a ^ b;

        System.out.println("c is : " + c);
```

*/

/* step 3 : For each keyword wk, the user creates an empty set swk=0.

For each i belongs to [1,n], if vwk[i] = 1, the user adds this file index i to the set swk.

In our code, since we have assumed that we have only one file, swk0 =1, swk1 = 1, swk2 = 1.

This means that certainly, all the three keywords k0, k1, k2 are present in the file F1 whose identity is 1.

* */

System.out.println("Step 3:");

Element swk0 = pairing.getZr().newElement(new BigInteger("1"));

Element swk1 = pairing.getZr().newElement(new BigInteger("1"));

Element swk2 = pairing.getZr().newElement(new BigInteger("1"));

System.out.println("swk0 is :" + swk0);

System.out.println("swk1 is :" + swk1);

System.out.println("swk2 is :" + swk2);

/* Step 4: For each keyword wk, the user computes the RAL

For each keyword wk, the user computes the RAL (Relation Authentication Label),

ohm(pi(wk)) = {ohm(wk,1), ohm(wk,2), ..., ohm(wk,s)}

Here, s refers to the numbers of blocks in a each file.

where, ohm(wk,1) = product of(H1(IDi||j)-1). H3(j). H2(pi(wk)||j)

In our project, we have assumed that, we have only 1 file.

we have three keywords in our project. k0, k1, k2.

k0 is present in file 1.

Similarly, k1, k2 are also in file 1.

In our project, we don't have more than one file, Hence,

all the keywords are present in this file 1.

*/

// Element ohm_wk1 = (product of(H1(ID1||1) inverse). H3(1). H2(pi(wk)||1)) power x

// Let us assume that,

// ohm_block1_v1 = product of(H1(ID1||1) inverse)

// ohm_block1_v2 = H3(1)

// ohm_block1_v3 = H2(pi(wk)||1)

// Therefore, Element ohm_wk1 = ohm_block1_v1.ohm_block1_v2.ohm_block1_v3

// Thus, since we have three keywords in our project, we have three values such as to compute

// ohm_wk1, ohm_wk2, ohm_wk3

System.out.println("Step 4:");

Element ohm_block1_v1 = pairing.getG1().newRandomElement().getImmutable();

Element ohm_block1_v2 = pairing.getG1().newRandomElement().getImmutable();

Element ohm_block1_v3 = pairing.getG1().newRandomElement().getImmutable();

Element ohm_wk1 = ((ohm_block1_v1.add(ohm_block1_v2)).add(ohm_block1_v3)).powZn(x);

Element ohm_block2_v1 = pairing.getG1().newRandomElement().getImmutable();

```
        Element ohm_block2_v2 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block2_v3 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_wk2 =
((ohm_block2_v1.add(ohm_block2_v2)).add(ohm_block2_v3)).powZn(x);



        Element ohm_block3_v1 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block3_v2 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block3_v3 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_wk3 =
((ohm_block3_v1.add(ohm_block3_v2)).add(ohm_block3_v3)).powZn(x);

        System.out.println("ohm_wk1 is " + ohm_wk1);

        System.out.println("ohm_wk2 is " + ohm_wk2);

        System.out.println("ohm_wk3 is " + ohm_wk3);

        bt3.addActionListener(new ActionListener() {
```

```
        @Override

        public void actionPerformed(ActionEvent e) {

            m1.setText("\nstep 1:"+"\npik0 is :"+pik0.toString()+"\npik1 is
:"+pik1.toString()+"\npik2 is :"+pik2.toString()+"\n\nstep 2:"+"\nevpiwko is
:"+evpiwk0.toString()+"\nevpiwk1 is :"+evpiwk1.toString()+"\nevpiwk2 is
:"+evpiwk2.toString()+"\n\nstep 3:"+"\nswk0 is :"+swk0.toString()+"\nswk1 is
:"+swk1.toString()+"\nswk2 is :"+swk2.toString()+"\n\nstep 4:"+"\nohm_wk1 is
:"+ohm_wk1.toString()+"\nohm_wk2 is :"+ohm_wk2.toString()+"\nohm_wk3 is
:"+ohm_wk3.toString());

            m1.setLineWrap(true);

        }

    });




    /* step 5 */

    // The user sets I = {pi(wk), evpiwk, ohm(pi(wk)} for each k=1,2,...,m

    /* Index gen - ends here */



    /* Authentication Generation - starts here */
```

/* For each encrypted block cij, the user computes the authenticator

sigma_ij = [H1(IDi||j).u power cij] power x

where, i = file no from 1 to n, j = block number from 1 to s

In our project,

we have only one file with identity 1, and 3 blocks.

Let us assume m1, m2, m3 represent the block values (which are integers).

Let us assume c1, c2, c3 represent the enctypted block values (which are integers).

Moreover, we have to compute the authenticators for only 3 blocks.

sigma_file1_block1, sigma_file1_block2, sigma_file1_block3

*/

System.out.println("\n\n 4.Authenticator Generation");

```
        Element sigma_file1_block1 =
((ohm_block1_v1.invert()).add(u.powZn(c1))).powZn(x);

        Element sigma_file1_block2 =
((ohm_block2_v1.invert()).add(u.powZn(c2))).powZn(x);

        Element sigma_file1_block3 =
((ohm_block3_v1.invert()).add(u.powZn(c3))).powZn(x);

        System.out.println("sigma_file1_block1 is:" + sigma_file1_block1);

        System.out.println("sigma_file1_block2 is:" + sigma_file1_block2);

        System.out.println("sigma_file1_block3 is:" + sigma_file1_block3);

        bt4.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m1.setText("\nsigma_file1_block1 is
:"+sigma_file1_block1.toString()+"\n\nsigma_file1_block2 is
:"+sigma_file1_block2.toString()+"\n\nsigma_file1_block3 is
:"+sigma_file1_block3.toString());

                m1.setLineWrap(true);

            }

        });
```

/* Authentication Generation - ends here */

/* trapdoor generation - starts here */

/*

The user computes the search trapdoor Tw' = { pi(w'), f(pi(w')) }

The trapdoor means, the data owner, i.e. the file owner whose is the owner of the file 1 in our case,

wants to search the file 1 using the keyword k0.

So, to securely send the keyword k0, he sends pik0, fpik0 to the TPA.

Apart from this, no value is computed in this module.

For trapdoor purpose, we call Tw as Tw_dash

T_w1_dash = {pik0, fpik0}

*/

System.out.println("\n\n 5.Trapdoor generation");

System.out.println("Let us assume that, we want search the blocks containing the keyword k0");

```
System.out.println("pik0 is : " + pik0);

System.out.println("fpik0 is : " + fpik0);

bt5.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        m1.setText("\nLet us assume that, we want search the blocks containing the keyword k0"+"\n\npik0 is :"+pik0.toString()+"\n\nfpik0 is :"+fpik0.toString());

        m1.setLineWrap(true);

    }

});

/* trapdoor generation - ends here */



/* Challenge generation - By TPA - starts here */

System.out.println("\n\n 6.Challenge Generation");

    /*

        Step 1: The TPA randomly chooses a c-elements subset Q from [1 to s]
```

In our project, we have s = 3. This means, we have only three blocks 1, 2, 3.

Let us assume that, we want to check the block no.1, block no. 2 only for verification.

Hence, our c-element subset has only block no.1, block no.2

i.e. c = {1,2}

*/

/*

Step 2: For each j in Q, the TPA randomly chooses vj from Zq*

In our case, j = 1, 2.

Therefore, the TPA selects, v1, v2 from Zq*

*/

```
Element v1 = pairing.getZr().newRandomElement().getImmutable();
Element v2 = pairing.getZr().newRandomElement().getImmutable();
System.out.println("v1 is : " + v1);
System.out.println("v2 is : " + v2);
```

```
bt6.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        m1.setText("\n\nv1 is :"+v1.toString()+"\n\nv2 is :"+v2.toString());
        m1.setLineWrap(true);
    }
});
    /* step 3: Now, the TPA sends the auditing challenge Chal = {Tw', {j, vj} j in Q} to the Cloud Server.
            In our case, Chal = {Tw0, {(1,v1), (2, v2)}}
     */
/* Challenge generation - ends here */

/* Proof Generation: By Cloud Server (CS)
    1. The CS parses the challenge Chal = {Tw0, {(1,v1), (2, v2)}}
    2. For the keyword pi(w) = pi(w'), the cloud find teh the corresponding
      encrypted row evpiwk and the RAL ohm_piwki from the secure index.
```

```
        The cloud decrypts the corresponding encrypted index vector

        vwk = evpiwk0 xor f(piwk)

    * */

    System.out.println("\n\n 7.Proof Generation");

    System.out.println("Step 1:");

    Element vw0 = pairing.getZr().newElement(new
BigInteger(element_xor(evpiwk0.toString(), fpik0.toString())));

    System.out.println("vw0 is : " + vw0);



    /* 3. The cloud initiates an empty set swk = phi. i.e. swk = {empty set}

        For each i in [1,n], if vwk[i] == 1, then the cloud add i to swk.



        In our case, we have i = 1 only. Here, i refers to the file no.

        Since, in our project, we have only one file, i =1 only.

        In our case, vwk[1] = 1. This is because, this keyword k0 is in the file 1.

        Therefore, we have to set swk0 = 1

    * *
```

```
 */

Element swk0_cs = pairing.getZr().newElement(new BigInteger("1"));

// This is same as setting swk0 in 3rd module, step c.



/* * Step 4: The cloud computes T = sigma_ij power vj for i in swk, j in Q

        In our project, we assume that, k0 is present in blocks m1, m2.

        Therefore, the authenticator's sigma_file1_block1,
sigma_file1_block2 for the encrypted blocks c1, c2 are taken by the CS.



 */

// Element T =
((sigma_file1_block1.powZn(v1)).add(ohm_wk1.powZn(v1))).add((sigma_file1_b
lock2.powZn(v2)).add(ohm_wk1.powZn(v2)));

Element T =
((sigma_file1_block1.powZn(v1)).add(sigma_file1_block2.powZn(v2))).add((ohm
_wk1.powZn(v1)).add(ohm_wk2.powZn(v2)));

Element meu = (c1.mul(v1)).add(c2.mul(v2));

System.out.println("Step 2 : ");
```

```
        System.out.println("T is : " + T);

        System.out.println("meu is : " + meu);

        bt7.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m1.setText("\nstep 1:"+"\nvw0 is :"+vw0.toString()+"\n\nStep 2 :"+"\nT
is :"+T.toString()+"\nmeu is :"+meu.toString());

                m1.setLineWrap(true);

            }

        });



        /*

            Now, the cloud server CS sets the auditing response

            Proof = {T, meu}

            That is, the CS sends this Proof to the TPA for verification.

         */
```

```
/* Proof generation - ends here */



/* Proof verification - starts here */

/* The TPA checks the validity of the following equation:

 *  e(T,g) = e( for j=1 in Q ((H3(j).H2(piw'||j ) power vj)).u power meu, y )

 *

 * In our project,

 * e(T,g) = e( ((H3(1).H2(piw')||1) power v1 + (H3(1).H2(piw')||2) power v2).u power meu, y)

 * Here, L.H.S. = e(T,g)

 *       R.H.S. = e( ((H3(1).H2(piw')||1) power v1 + (H3(1).H2(piw')||2) power v2).u power meu, y)

 */

System.out.println("\n\n 8.Proof Verification");

Element LHS = pairing.pairing(T, g);

System.out.println("LHS is :" + LHS);

// ohm_block1_v1 = product of(H1(ID1||1) inverse)
```

```
        // ohm_block1_v2 =  H3(1)

        // ohm_block1_v3 =  H2(pi(wk)||1)

        Element RHS =
pairing.pairing((((ohm_block1_v2.add(ohm_block1_v3)).powZn(v1)).add((ohm_bl
ock2_v2.add(ohm_block2_v3)).powZn(v2))).add(u.powZn(meu)), y);

        System.out.println("RHS is :" + RHS);

        bt8.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m1.setText("\nLHS is :"+LHS.toString()+"\n\nRHS is
:"+RHS.toString());

                m1.setLineWrap(true);

            }

        });



        /* Proof verification - ends here */



        /* Only for rough use
```

```
    Element p1 = pairing.getG1().newRandomElement().getImmutable();
    Element p2 = p1.invert().getImmutable();
    System.out.println("\n\n For Rough use only");


    System.out.println("p1 is : " + p1);
    System.out.println("p2 is : " + p2);
    Element p3 = p1.sub(p2);
    System.out.println("p3 is : " + p3);
     Only for rough use */
}


public static String element_xor(String v1, String v2) {
    System.out.println("v1 is : " + v1);
    System.out.println("v2 is : " + v2);
    StringBuffer v3 = new StringBuffer();
    for (int i = 0; i < v1.length(); i++) {
```

```
            v3.append(v1.charAt(i) ^ v2.charAt(i));
        }
        System.out.println("v3 is : " + v3);
        return v3.toString();
// Code ends here


    }
    public static void main(String[] args) {
        new Main();
    }
}
```